**CCM5 - ENTREGA E APRESENTAÇÃO EM 28/05 e 04/06**

**Grupo de até 4 pessoas**

**Artigo (no mínimo 3 páginas) – 3.0 pontos**

* Introdução com os principais conceitos e fundamentos
* explicação das técnicas / algoritmo aplicados
* Descritivo dos resultados, de preferencia mostrando figuras didáticas sobre o funcionamento da técnica
* Bibliografia utilizada

Artigos que incluírem o código do programa e mal dão alguma explicação para o algoritmo geralmente recebem notas que não são legais. :(

Artigos baseados em GPT são geralmente detectados, e a nota é algo bemmmm ruim.

**Programa fonte funcional - 3.0 pontos – Linguagem C ou python**

* Deve ter interface gráfica e boa interação com o usuário
* Bibliotecas somente podem ser usadas para interface com o usuário, nunca para a implementação dos pontos requeridos em cada trabalho

**Apresentação ( 20 a 30 min) - 4.0 pontos**

**Trabalhos**

1. Implementar um **compactador de Huffman**.
   1. O mesmo deve permitir ao usuário escolher um arquivo txt e **compactar** o mesmo.
   2. Deve permitir a **descompactação** do arquivo, retornando ao texto original
   3. Deve mostrar o arquivo compactado (pode ser um slide na apresentação)
   4. **Deve mostrar a árvore de huffman montada.**
   5. Deve mostrar e explicar a taxa de compactação (pode ser um slide na apresentação).
   6. É possível (e desejável) combinar huffman com outras técnicas para melhorar a performance
   7. **Um arquivo padrão para compactação/decompactação será fornecido no dia da apresentação e será usado como métrica de sucesso.**
2. Implementar o algoritmo de **Dijkstra**
   1. O usuário deve poder criar o grafo **direcionado** e ponderado
   2. Após a criação, o usuário pode escolher um vértice inicial e o algoritmo deve calcular o custo para todos os vértices
   3. Após calcular o custo, o algoritmo deve mostrar ao usuário a rota que oferece o menor custo
3. Implementar um algoritmo para criar uma arvore **geradora mínima usando Kruskal**
   1. O algoritmo deve receber um **grafo não dirigido e valorado**, e criar uma arvore **geradora mínima**.
   2. O algoritmo deve informar ao usuário quantos circuitos foram retirados do grafo original.
   3. Deve mostrar os circuitos retirados
   4. O algoritmo deve informar quando a arvore geradora mínima é o próprio grafo
   5. O algoritmo deve mostrar ao usuário a formação final da arvore, com os respectivos custos de suas arestas
4. Implementar um algoritmo que receba um grafo **DIRECIONADO** através de uma matriz de adjacência e informe:
   1. Se o grafo tem um ciclo e/ou caminho Eureliano
   2. Se o grafo tem um ciclo e/ou caminho hamiltoniano
   3. Mostre os ciclos caminhos Eurelianos e hamiltonianos encontrados
   4. A procura por ciclos / caminhos hamiltonianos devem permitir procuras heurísticas ou exata.
   5. A procura deve incluir um limite de tempo e/ou de vértices percorridos.